Департамент образования города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

Башашкин Алексей Максимович

Лабораторная работа по «Инструменты для хранения и обработки больших данных»

Выполнил:
Башашкин Алексей Максимович
Проверил:
Босенко Тимур Муртазович

Курс обучения: 4
Форма обучения: очная

Москва
2025

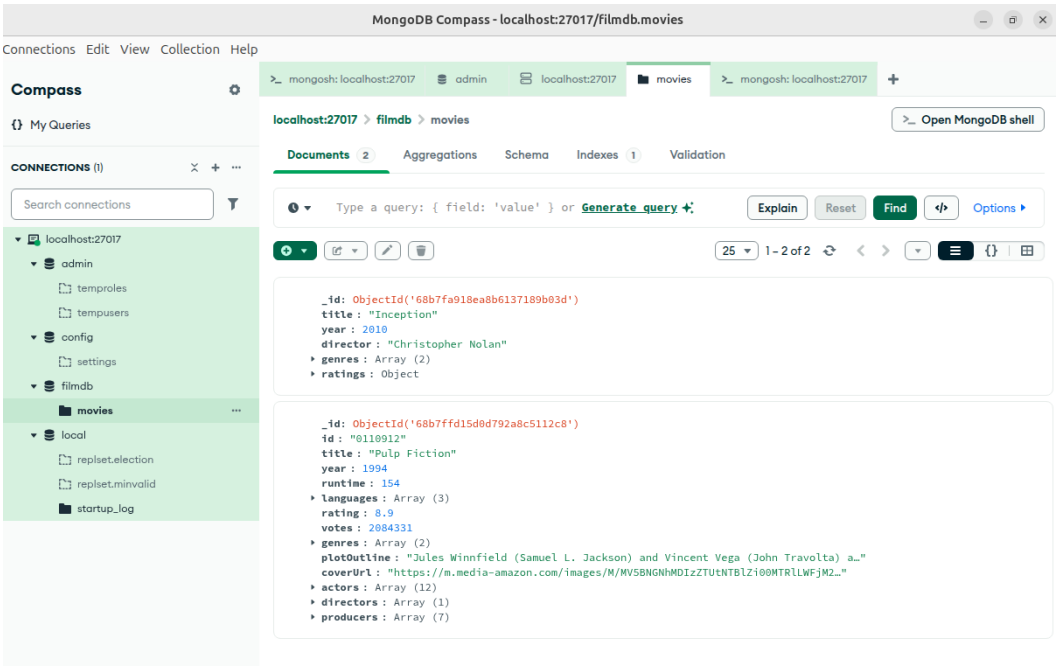
Цель работы: изучить работу трех различных NoSQL баз данных: документо-ориентированной (MongoDB), графовой (Neo4j) и хранилищем типа "ключ-значение" (Redis). Сравнить синтаксис и подходы к формированию запросов в различных NoSQL-системах.

Ход работы

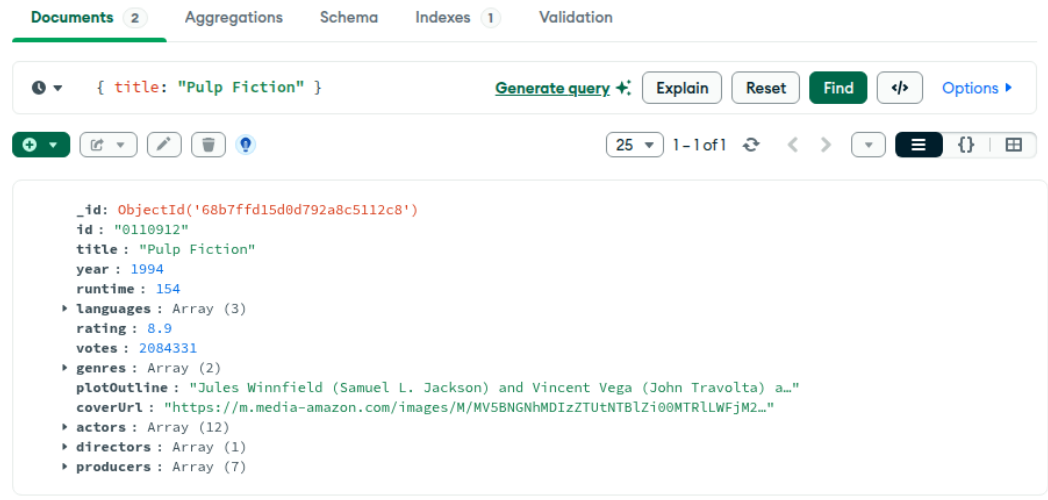
Запуск контейнера

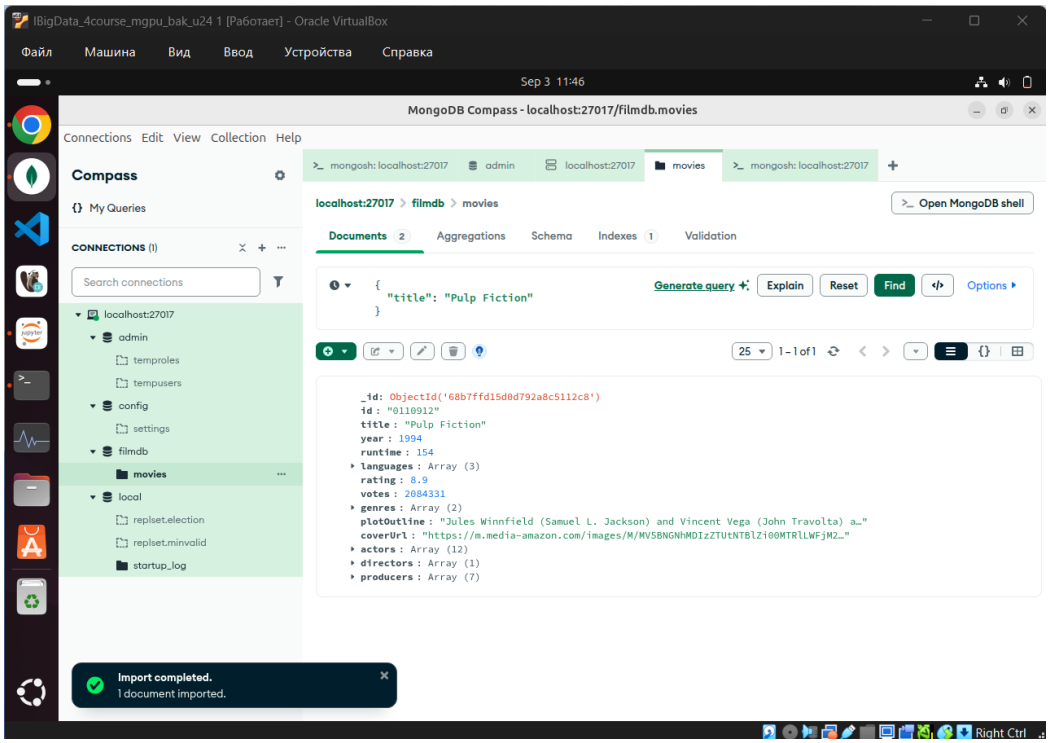
```
mgpu@mgpu-vm:~/Downloads/ibd/nosql-workshop/01-environment$ cd docker
mgpu@mgpu-vm:~/Downloads/ibd/nosql-workshop/01-environment/docker$ sudo docker c
ompose up -d
[+] Running 10/10
 ✔ Network nosql-platform      Created                                    0.3s
 ✔ Container redis-commander   Started                                    4.6s
 ✔ Container cassandra-web     Started                                    3.8s
 ✔ Container cassandra-1       Started                                    3.6s
 ✔ Container mongo-1           Started                                    4.9s
 ✔ Container admin-mongo       Started                                    5.0s
 ✔ Container redis-1           Started                                    6.3s
 ✔ Container jupyter           Started                                    4.4s
 ✔ Container mongo-express     Started                                    6.5s
 ✔ Container neo4j-1           Started                                    5.7s
mgpu@mgpu-vm:~/Downloads/ibd/nosql-workshop/01-environment/docker$ docker exec -
it mongo-1 mongosh -u mongoroot -p mongopass --authenticationDatabase admin
Current Mongosh Log ID: 68b7f99eb51fa43c4489b03c
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnectio
```

Интерфейс MongoDB Compass



Поиск в документе





Команда find() возвращает список документов из БД

```
> db.getCollectionNames()
< [ 'movies' ]
> db.movies.find()
< {
    _id: ObjectId('68b7fa918ea8b6137189b03d'),
    title: 'Inception',
    year: 2010,
    director: 'Christopher Nolan',
    genres: [
      'Sci-Fi',
      'Action'
    ],
    ratings: {
      imdb: 8.8,
      metacritic: 74
    }
  }
```

Добавление актеров в коллекцию persons

```
db.persons.insertOne (
{
    "id": 0000246,
    "name": "Bruce Willis",
    "headshot": "https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_C
    "birthDate": "1955-03-19",
```

```
db.persons.insertOne (
{
    "id": 0000206,
    "name": "Keanu Reeves",
    "headshot": "https://m.media-amazon.com/images/M/MV5BMjA0MjMzMTE5OF5BMl5BanBnXkFtZTcwMzQ2ODE3Mw@@._V1_UY98_C
```

```
db.persons.insertOne (
{
    "id": 0000113,
    "name": "Sandra Bullock",
    "headshot": "https://m.media-amazon.com/images/M/MV5BMTI5NDY5NjU3NF5BMl5BanBnXkFtZTcwMzQ0MTMyMw@@._V1_UX67_C
```

```
db.persons.insertOne (
{
    "id": 0000233,
    "name": "Quentin Tarantino",
    "headshot": "https://m.media-amazon.com/images/M/MV5BMTgyMjI3ODA3Nl5BMl5BanBnXkFtZTcwNzY2MDYxOQ@@._V1_UX67_C
```

Убедимся, что все 4 актера были добавлены в коллекцию

```
> db.persons.countDocuments()
< 4
filmdb >
```

Вывод всех фильмов жанра Family

```
> db.movies.find({"genres": "Family"})
< {
    _id: ObjectId('68bf61b6de97a3091491a14e'),
    id: '0038650',
    title: "It's a Wonderful Life",
    genres: [
      'Drama',
      'Family',
      'Fantasy'
```

Вывод фильмов, которые не относятся к жанру «Драма»

```
> db.movies.find({"genres": { $ne: "Drama"} })
< {
    _id: ObjectId('68b7fa918ea8b6137189b03d'),
    title: 'Inception',
    year: 2010,
    director: 'Christopher Nolan',
    genres: [
      'Sci-Fi',
      'Action'
    ],
```

Оператор $exists проверяет на наличие или отсутствие поля. Так, можно проверить, есть ли у фильма краткое изложение сюжета

```
> db.movies.find({ "plotOutline": { $exists: true} })
< {
    _id: ObjectId('68b7ffd15d0d792a8c5112c8'),
    id: '0110912',
    title: 'Pulp Fiction',
    year: 1994,
    runtime: 154,
    languages: [
      'en',
      'es',
      'fr'
    ],
    rating: 8.9,
    votes: 2084331,
    genres: [
      'Crime',
      'Drama'
    ],
    plotOutline: 'Jules Winnfield (Samuel L. Jackson) and Vincent Vega
    coverUrl: 'https://m.media-amazon.com/images/M/MV5BNGNhMDIzZTUtNTBl
    actors: [
      {
```

Также поддерживаются логические операторы OR, AND. Например, в этом запросе выводятся фильмы жанра «Экшн» и позже 2010 года или фильмы с рейтингом выше 8.8

```
> db.movies.find({ "genres":"Action", $or: [ { "year": { $gte :  2010 } },  { "rating": { $gt :  8.8 } } ] })
< {
    _id: ObjectId('68b7fa918ea8b6137189b03d'),
    title: 'Inception',
    year: 2010,
    director: 'Christopher Nolan',
    genres: [
      'Sci-Fi',
      'Action'
    ],
    ratings: {
      imdb: 8.8,
      metacritic: 74
    }
  }
  {
```

Обновление документов

```
> db.movies.updateOne ( {title: 'Fight Club'} , { $set: {rating: 9} } )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

Оптимизация запросов с помощью индексов

Создание индекса

```
> db.movies.createIndex( {title: 1} );
< title_1
```

Вывод плана запроса

```
> db.movies.find ( {title: "The Matrix"} ).explain();
< {
    explainVersion: '1',
    queryPlanner: {
      namespace: 'filmdb.movies',
      indexFilterSet: false,
      parsedQuery: {
        title: {
          '$eq': 'The Matrix'
        }
      },
      queryHash: '553F28EB',
      planCacheKey: 'DA190D56',
      optimizationTimeMillis: 0,
      maxIndexedOrSolutionsReached: false,
      maxIndexedAndSolutionsReached: false,
      maxScansToExplodeReached: false,
      winningPlan: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: {
            title: 1
```

Создание текстового индекса для поиска текста, если он встречается в названии или кратком изложении сюжета фильма

```
> db.movies.createIndex ( { title: "text", plotOutline: "text" } )
< title_text_plotOutline_text
> db.movies.find( { $text: { $search: "fight" } } )
< {
    _id: ObjectId('68bf61b6de97a3091491a140'),
    id: '0137523',
    title: 'Fight Club',
    genres: [
      'Drama'
    ],
    year: 1999,
    rating: 9,
    rank: 10
  }
  {
    _id: ObjectId('68b7ffd15d0d792a8c5112c8'),
    id: '0110912',
    title: 'Pulp Fiction',
```

Агрегация данных

Вывод количества фильмов по рейтингу

```
> db.movies.aggregate( [{$group:{_id:'$rating', total: { $sum:1 }}}])
< {
    _id: 8.6,
    total: 7
  }
  {
    _id: 8.8,
    total: 3
  }
  {
    _id: 9.2,
    total: 2
  }
  {
    _id: 8.4,
    total: 1
```
0

Вывод количества, минимальной, максимальной и средней оценки фильмов позже 2000 года по жанру. Результат отсортирован по количеству фильмов в жанре в порядке убывания

```
> db.movies.aggregate([
                    {$match: {year:{$gt:2000}}},
                    {$unwind: "$genres" },
                    {$group: {_id:'$genres',
                        number :{ $sum:1 },
                        minRating:{$min:'$rating'},
                        maxRating:{$max:'$rating'},
                        avgRating:{$avg:'$rating'}
                    }},
                    {$sort:{number:-1}} ])
< {
    _id: 'Drama',
    number: 11,
    minRating: 8.5,
    maxRating: 9,
    avgRating: 8.636363636363637
  }
  {
    _id: 'Adventure',
    number: 7,
    minRating: 8.5,
    maxRating: 8.9,
    avgRating: 8.7
  }
```

Удаление документов

```
> db.movies.deleteOne( { "title": "Fight Club" } )
< {
    acknowledged: true,
    deletedCount: 1
  }
```

Redis

```
SET server:name "redis-server"
"OK"
GET server:name
"redis-server"
SET connections 10
"OK"
GET connections
"10"
SET connections 20
"OK"
GET connections
"20"
SETNX connections 30
0
GET connections
"20"
SETNX newkey 30
1
MSET key1 10 key2 20 key3 30
"OK"
```

SET - Сохранение значения «redis-server» по ключу «server:name»

GET - Получение значения по ключу «server:name»

SETNX – Создает значение для ключа, если этот ключ не существует

MSET - Установка нескольких пар «ключ-значение»

MGET – Возращение нескольких значений для нескольких ключей

```
MGET key1 key3
1) "10"
2) "30"
SET connections 10
"OK"
INCR connections
11
INCRBY connections 10
21
DECR connections
20
DECRBY connections 10
10
DEL connections
1
EXISTS connections
0
INCR connections
1
SET resource:lock "Redis Demo"
"OK"
```

INCR – Операция инкремента

INCRBY – Операция инкремента на x

DECR / DECRBY – Операции декремента

DEL – Удаление пары «ключ-значение»

```
EXPIRE resource:lock 120
1
TTL resource:lock
99
TTL resource:lock
15
TTL resource:lock
-2
EXISTS resource:lock
0
SET resource:lock "Redis Demo 1" EX 120
"OK"
TTL resource:lock
114
SET resource:lock "Redis Demo 2"
"OK"
TTL resource:lock
-1
RPUSH skills "Oracle RDBMS"
1
```

EXPIRE - Установка срока действия пары «ключ-значение»

TTL - Проверка срока действия пары «ключ-значение». Если возвращает -2, то это значит, что ключ не существует

```
RPUSH skills "Redis"
2
GET skills
"WRONGTYPE Operation against a key holding the wrong kind of value"
LRANGE skills 0 -1
1) "Oracle RDBMS"
2) "Redis"
LPUSH skills "SQL Server"
3
LRANGE skills 0 -1
1) "SQL Server"
2) "Oracle RDBMS"
3) "Redis"
LLEN skills
3
LPOP skills
"SQL Server"
RPOP skills
"Redis"
```

RPUSH - Добавление нового элемента в конец списка

LPUSH - Добавление элемента в начало списка

LRANGE - Возвращение элементов списка. Диапазон 0 -1 значит от начала списка до конца.

LLEN – Возвращение длины списка

LPOP - Удаление и возвращение первого элемента списка

RPOP – Удаление и возвращение последнего элемента списка

```
LLEN skills
1
LRANGE skills 0 -1
1) "Oracle RDBMS"
SADD nosql:products "Cassandra"
1
SADD nosql:products "Redis"
1
SADD nosql:products "MongoDB"
1
SMEMBERS nosql:products
1) "Cassandra"
2) "Redis"
3) "MongoDB"
SREM nosql:products "MongoDB"
1
SMEMBERS nosql:products
1) "Cassandra"
2) "Redis"
SISMEMBER nosql:products "Cassandra"
1
SISMEMBER nosql:products "MongoDB"
0
SADD rdbms:products "Oracle"
```

SADD – Добавление значения в множество

SMEMBERS – Возвращение списка всех элементов множества

SREM – Удаление значения из множества

SISMEMBER – Возвращает 1, если элемент существует в множестве, и 0 - если нет.

```
SUNION rdbms:products nosql:products
1) "Oracle"
2) "SQL Server"
3) "Cassandra"
4) "Redis"
SUNIONSTORE database:products rdbms:products nosql:products
4
SMEMBERS database:products
1) "Oracle"
2) "SQL Server"
3) "Cassandra"
4) "Redis"
SADD favorite:products "Cassandra"
1
SADD favorite:products "Oracle"
1
SINTER database:products favorite:products
1) "Cassandra"
2) "Oracle"
ZADD pioneers 1940 "Alan Kay"
1
ZADD pioneers 1906 "Grace Hopper"
1
ZADD pioneers 1953 "Richard Stallman"
1
ZADD pioneers 1965 "Yukihiro Matsumoto"
1
```

SUNION – Объединение двух множеств

SUNIONSTORE – Объединение двух или нескольких множеств

SINTER – Возвращает список пересекающихся элементов из нескольких множеств

ZADD – Добавление одного или нескольких элементов в упорядоченное множество

```
ZRANGE pioneers 2 4
1) "Claude Shannon"
2) "Alan Kay"
3) "Richard Stallman"
ZRANGE pioneers 2 4 WITHSCORES
1) "Claude Shannon"
2) "1916"
3) "Alan Kay"
4) "1940"
5) "Richard Stallman"
6) "1953"
ZREVRANGE pioneers 0 2
1) "Linus Torvalds"
2) "Yukihiro Matsumoto"
3) "Sophie Wilson"
HSET user:1000 name "John Smith"
1
HSET user:1000 email "john.smith@example.com"
1
HSET user:1000 password "s3cret"
1
```

ZRANGE – Возращение диапазона элементов в упорядоченном множестве по индексу (отсортировано по возрастанию). WITHSCORES возвращает значения этих элементов.

ZREVRANGE – Аналогично ZRANGE, но отсортировано по убыванию

HSET – Установка значения для поля в хэше, хранящемся по заданному ключу

```
HGETALL user:1000
{
  "name": "John Smith",
  "email": "john.smith@example.com",
  "password": "s3cret"
}
HMSET user:1001 name "Mary Jones" password "hidden" email "mjones@example.com"
"OK"
HGETALL user:1001
{
  "name": "Mary Jones",
  "password": "hidden",
  "email": "mjones@example.com"
}
HGET user:1001 name
"Mary Jones"
HSET user:1000 visits 10
1
HINCRBY user:1000 visits 1
11
HINCRBY user:1000 visits 10
21
HDEL user:1000 visits
1
GEOADD cities:russia 37.6176 55.7558 "Moscow"
1
```

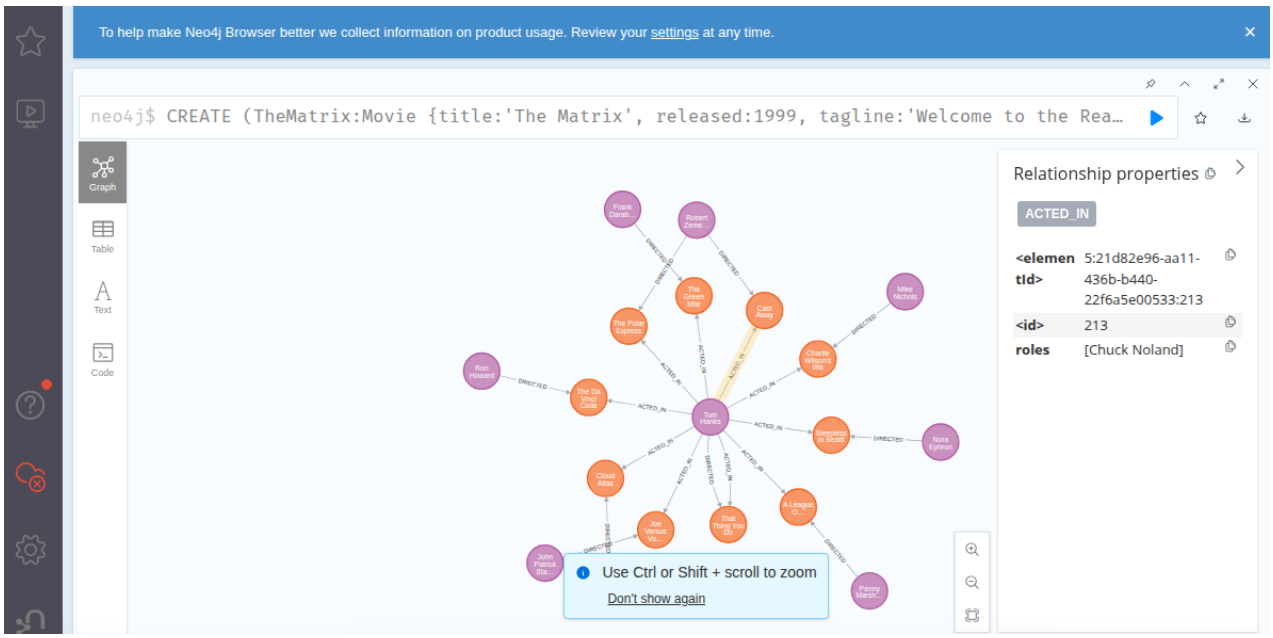
HGETALL – Возравщение сохраненных данных

HMSET – Установка нескольких полей одновременно

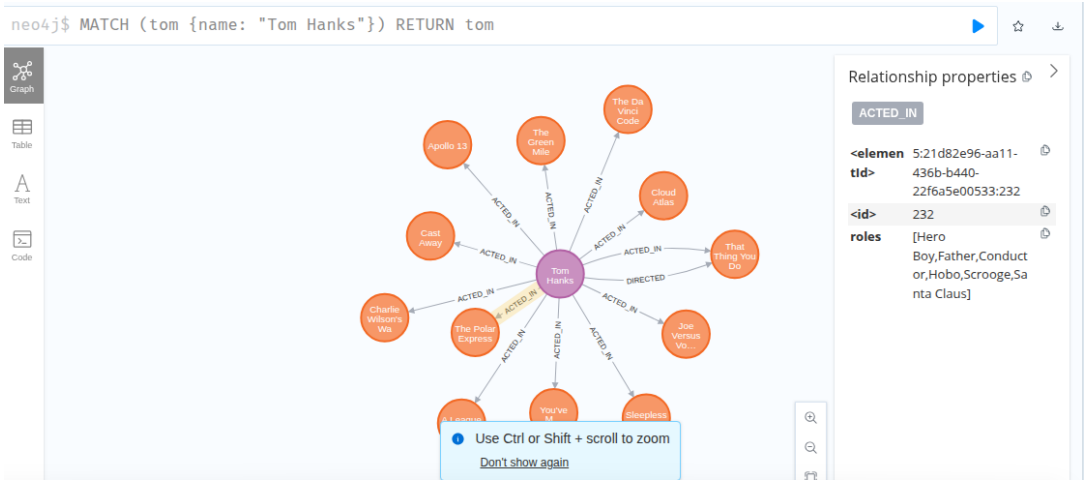
HGET – Возвращение значений одного поля

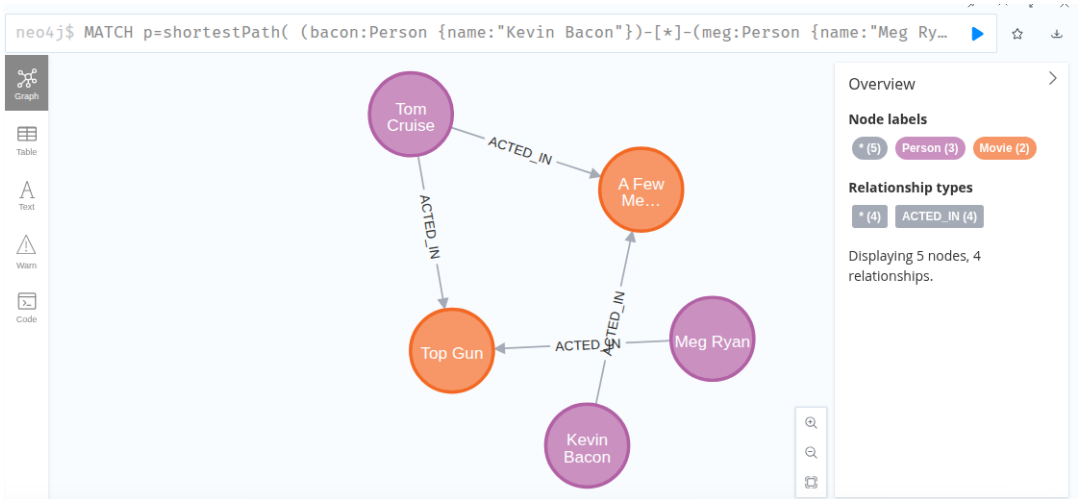
Neo4j

Создание БД



Вывод всех данных Тома Хенкса





Индивидуальное задание

Вариант 5

1) Постановка задачи: найти все фильмы, которые были выпущены до 1980 года ИЛИ имеют рейтинг выше 8.9 ($or). Отсортировать результат по году в порядке убывания.

Код запроса:

```
db.movies.find({
  $or: [
    { year: { $lt: 1980 } },
    { rating: { $gt: 8.9 } }
  ]
}).sort({ year: -1 })
```

Результат:

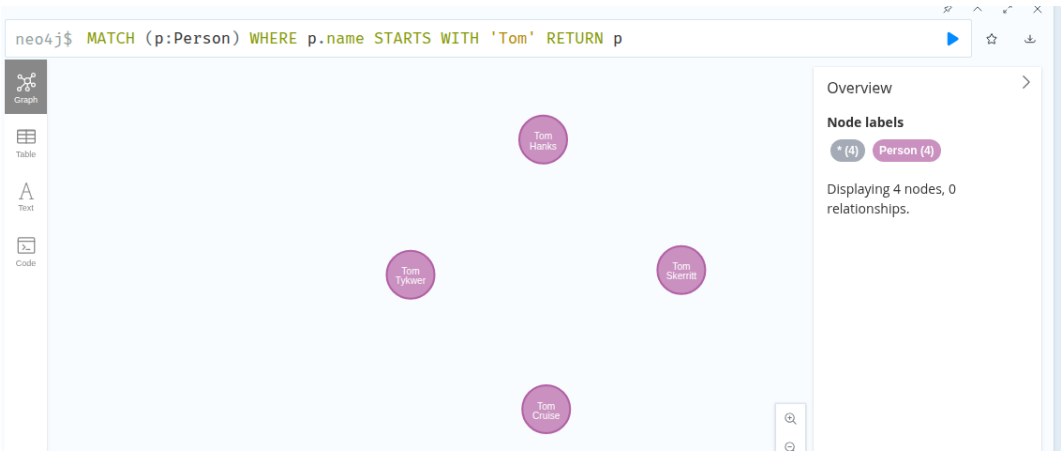
2) Постановка задачи: найти всех людей (актеров или режиссеров), чье имя начинается на "Tom".

Код запроса: MATCH (p:Person)

WHERE p.name STARTS WITH 'Tom'

RETURN p

Результат:



3) Постановка задачи: смоделировать таблицу лидеров: с помощью **упорядоченного множества** (ZADD) leaderboard:game1 добавить 4 игроков с их очками. Увеличить счет одного из игроков на 100 очков (ZINCRBY).

Код задачи: ZADD leaderboard:game1 2500 "player1" 1800 "player2" 3200 "player3" 1500 "player4"

ZINCRBY leaderboard:game1 100 "player2"

Результат:

```
ZADD leaderboard:game1 2500 "player1" 1800 "player2" 3200 "player3" 1500 "player4"
4
ZINCRBY leaderboard:game1 100 "player2"
"1900"
ZREVRANGE leaderboard:game1 0 -1 WITHSCORES
1) "player3"
2) "3200"
3) "player1"
4) "2500"
5) "player2"
6) "1900"
7) "player4"
8) "1500"
```

Выводы: В ходе работы освоены ключевые концепции трех NoSQL-СУБД: документная модель MongoDB с коллекциями и документами, графовая модель Neo4j с поиском по шаблонам и обходом связей, а также ключ-значная модель Redis с упорядоченными множествами и атомарными операциями. Основные трудности с синтаксическими различиями и парадигмальным переключением между системами были преодолены через практическую работу с документацией. Приобретены навыки выбора подходящей СУБД под конкретные задачи, проектирования структур данных и выполнения базовых операций в каждой из изученных систем.